WOODCOCK WASHBURN

INTELLECTUAL PROPERTY LAW

ATLANTA • PHILADELPHIA • SEATTLE

FACSIMILE

JOB CODE:

DATE: **March 16, 2009**

OFFICIAL PAPER

Please deliver this and the following pages to:

Examiner:	**Ben C. Wang**
U.S.P.T.O. Group Art Unit:	**2192**
Telecopier No.:	**571-270-2240**
U.S. Serial No.:	**10/676,743**
Client/Matter No.:	**MSFT-2763/305222.1**
Sender's Name:	**Joseph F. Oriti**
Pages to Follow:	**17**

If transmission is not complete, please call our **Philadelphia Office** at **(215) 568-3100**.

COVER MESSAGE:

OFFICIAL FACSIMILE. PLEASE DELIVER TO EXAMINER IMMEDIATELY:

Per our discussion, here is a proposed draft office action response for our telephone conversation to be scheduled for March 25, 2009 at 4:00 p.m.

DOCKET NO.: MSFT-2763/305222.1 **PATENT**
Application No.: 10/676,743
Office Action Dated: January 16, 2009

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:
Anandhi Somasekaran, Sergey Chub, Tolga Yildirim, Gueorgui Bonov Chkodrov, Kraig S. Rury, Lucy Ling-Chu Chao, Vladimir Pogrebinsky

Confirmation No.: **7938**

Application No.: **10/676,743**

Group Art Unit: **2192**

Filing Date: **September 30, 2003**

Examiner: **Ben C. Wang**

For: **Non-Disruptive Business Process Debugging And Analysis**

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

REPLY PURSUANT TO 37 CFR § 1.111

In response to the Official Action dated **January 16, 2009**, reconsideration is respectfully requested in view of the amendments and/or remarks as indicated below:

☐ **Amendments to the Specification** begin on page of this paper.

☒ **Amendments to the Claims** are reflected in the listing of the claims which begins on page 2 of this paper.

☐ **Amendments to the Drawings** begin on page of this paper and include an attached replacement sheet.

☒ **Remarks** begin on page 113 of this paper.

DOCKET NO.: MSFT-2763/305222.1 **PATENT**
Application No.: 10/676,743
Office Action Dated: January 16, 2009

This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A business process service debugger for remotely debugging a business process service, comprising:

means for indirectly communicating with a remote computer implementing the business process service, permitting the debugger to remotely debug the business process service without prior knowledge what remote computer it is running on;

means for establishing a communications connection with a remote computer~~, wherein the remote computer is implementing the business process service~~;

means for reading stored state information regarding historical events related to at least one business process implemented for the business process service;

means for displaying a symbolic representation of the operation of the business process service based on the stored state information; and

means for remotely debugging the business process service using the symbolic representation, communications connection and stored state information.

2. (Previously Presented) The business process service debugger of claim 1, wherein business processes and instances of the business process service other than those being debugged are not disrupted during debugging.

3. (Previously Presented) The business process service debugger of claim 1, wherein the symbolic representation comprises a workflow of at least one business process in the business process service.

4. (Previously Presented) The business process service debugger of claim 1, further comprising means for interacting with the business process service according to a user instruction.

5. (Original) The business process service debugger of claim 1, wherein the stored state information corresponds to a variable assignment within the business process service.

6. (Previously Presented) The business process service debugger of claim 1, wherein the events are historical events that occurred prior to failure of the at least one business process.

7. (Currently Amended) The business process service debugger of claim 1, wherein the stored state information corresponds to message flow data and an order in which run time components performed the at least one business process as a result of message processing.

8. (Original) The business process service debugger of claim 1, wherein said reading means further comprises means for reading stored business process service configuration information.

9. (Previously Presented) The business process service debugger of claim 1, wherein the events are events that occur prior to an inserted breakpoint in the one business process.

10. (Original) The business process service debugger of claim 1, wherein said debugging means comprises means for detecting a location where the instance is being processed.

11. (Original) The business process service debugger of claim 1, wherein said debugging means comprises means for detecting a location where the instance state is being stored.

12. (Currently Amended) A system for remotely debugging a distributed transactional application, comprising:

a server configured to execute an instance of a business process service comprising a plurality of business processes, thereby generating runtime data;

a client computer configured to execute a debugging user interface (UI) process that indirectly communicates with the server, establishes a communications connection with the

server, requests historical runtime data for at least one of the plurality of business processes, and generates, based on the historical runtime data, a symbolic representation of the business service process showing any debugging break points specified by a user; and

an interceptor for monitoring the runtime data and, when a specified break point is identified, causing the server to enter or leave a debugging state.

13. (Original) The system of claim 12, further comprising a database for receiving the runtime data and for storing business process service state information.

14. (Previously Presented) The system of claim 13, further comprising a display device for displaying the symbolic representation, and a user input device, wherein the input device is used to specify debugging break points.

15. (Previously Presented) The system of claim 14, wherein the symbolic representation comprises a workflow representative of the program flow of the business process service.

16. (Previously Presented) The system of claim 14, wherein the display device further displays data representative of a message flow of the business process service.

17. (Previously Presented) The system of claim 14, wherein the symbolic representation is presented according to stored state information.

18. (Previously Presented) The system of claim 12, wherein a message box database is coupled between the server and client computer and is configured for communicating debugging requests from the client computer.

19. (Previously Presented) The system of claim 18, wherein the UI process comprises an application program interface for communicating with the message box database.

20. (Previously Presented) The system of claim 18, further comprising a tracking database to receive business process service tracking information, wherein the UI process comprises a UI component for communicating with the tracking database.

21. (Canceled)

22. (Canceled)

23. (Original) The system of claim 12, wherein the interceptor is a component of a computer language that provides stored state tracking information.

24. (Original) The system of claim 12, wherein the UI process detects a location where the instance is being processed.

25. (Original) The system of claim 12, wherein the UI process detects a location where the instance state is being stored.

26. (Previously Presented) A method for debugging an instance of a business process service running on a remote computer, comprising:

generating for display, in a graphical user interface (GUI), a symbolic representation of the business process service based on a correlation of information about the design and execution of the business process service;

receiving a debugging command generated by a user interacting with the GUI;

establishing a direct client connection channel with the remote computer;

causing an interceptor to monitor runtime data generated by the instance of the business process service in accordance with the debugging command;

receiving a runtime request, generated by a user interacting with the GUI, for event information generated by execution of the instance of the business process service;

sending the runtime request to the remote computer for processing by the remote computer.

27. (Original) The method of claim 26, further comprising:

querying a database containing a status of the business process service;

displaying a query result on a display device;

receiving user input with respect to the query result; and

establishing the direct client connection channel in response to the user input.

28. (Original) The method of claim 27, wherein the information contained in the database is instance runtime data.

29. (Original) The method of claim 27, wherein the information contained in the database is instance tracking data.

30. (Previously Presented) The method of claim 26, further comprising:

creating the business process service using a process designer;

saving a business process service configuration and symbolic data in a database as information about the design of the business process service;

displaying the symbolic representation on a display device according to the saved business process service configuration and symbolic data;

generating a runtime request based on the symbolic representation; and

displaying a result of the runtime request on the display device.

31. (Previously Presented) The method of claim 30, wherein the symbolic representation comprises a shape corresponding to an operation in the business process service.

32. (Previously Presented) The method of claim 30, wherein the symbolic representation comprises a workflow representation of the business process service.

33. (Original) The method of claim 30, wherein the saving step takes place in connection with compiling and deploying the business process service.

34. (Original) The method of claim 30, wherein the business process service is implemented in a computer language that provides stored state information.

35. (Previously Presented) The method of claim 26, wherein the debugging command is a break point.

36. (Canceled)

37. (Previously Presented) The method of claim 26, wherein the runtime data is state information.

38. (Original) The method of claim 26, further comprising detecting a location where the instance is being processed.

39. (Original) The method of claim 26, further comprising detecting a location where an instance state is being stored.

40. (Previously Presented) A method in a computer system for displaying on a display device a graphical debugging environment for a business process service, the method comprising:

obtaining design information about the business process service;

obtaining tracking information about execution of the business process service;

generating a symbolic representation of the operation of the business process service according to the design information and tracking information; and

displaying on the display device a graphical debugging environment showing the symbolic representation.

41. (Original) The method of claim 40, further comprising receiving runtime data for the business process service and presenting the runtime data on the display device.

42. (Previously Presented) The method of claim 41, wherein the runtime data comprises historical message flow information including identification of run time messages that were constructed as a result of processing received messages, and further comprises order information pertaining to the order in which different run time components were executed as a result of processing received messages.

43. (Previously Presented) The method of claim 40, wherein the graphical debugging environment enables a user to place a breakpoint in the symbolic representation of the operation of the business process service.

44. (Previously Presented) The method of claim 40, the symbolic representation comprising symbols, wherein the graphical debugging environment also displays information about the symbols.

45. (Previously Presented) The method of claim 40, further comprising receiving input from an input device to place a break point proximate a symbol, and presenting a symbol representing the break point on the symbolic representation.

46. (Previously Presented) A computer-readable storage medium having computer-executable instructions for performing a method for debugging an instance of a business process service running on a remote computer, comprising:

generating for display, in a graphical user interface (GUI), a symbolic representation of the business process service based on a correlation of information about the design and execution of the business process service;

receiving a debugging command generated by a user interacting with the GUI;

establishing a direct client connection channel with the remote computer;

causing an interceptor to monitor runtime data generated by the instance of the business process service in accordance with the debugging command;

receiving a runtime request; and

sending the runtime request to the remote computer for processing by the remote computer.

47. (Previously Presented) The computer-readable storage medium of claim 46, wherein the method further comprises:

querying a database containing a status of the business process service;

displaying a query result on a display device;

receiving user input with respect to the query result; and

establishing the direct client connection channel in response to the user input.

48. (Previously Presented) The computer-readable storage medium of claim 47, wherein the information contained in the database is instance runtime data.

49. (Previously Presented) The computer-readable storage medium of claim 47, wherein the information contained in the database is instance tracking data.

50. (Previously Presented) The computer-readable storage medium of claim 46, wherein the method further comprises:

creating the business process service using a process designer;

saving business process service configuration data in a database as information about the design of the business process service;

displaying the symbolic representation on a display device according to the saved business process service configuration data;

generating a runtime request based on the symbolic representation; and

displaying a result of the runtime request on the display device.

51. (Previously Presented) The computer-readable storage medium of claim 50, wherein the symbolic representation comprises a shape corresponding to an operation in the business process service.

52. (Previously Presented) The computer-readable storage medium of claim 50, wherein the symbolic comprises a workflow representation of the business process service.

53. (Previously Presented) The computer-readable storage medium of claim 50, wherein the saving step takes place in connection with compiling and deploying the business process service.

54. (Previously Presented) The computer-readable storage medium of claim 50, wherein the business process service is implemented in a computer language that provides stored state information.

55. (Previously Presented) The computer-readable storage medium of claim 50, wherein the debugging command is a break point.

56. (Previously Presented) The computer-readable storage medium of claim 50, wherein the debugging command is a request for data regarding an instance of the business process service.

57. (Previously Presented) The computer-readable storage medium of claim 56, wherein the runtime data is state information.

58. (Previously Presented) The computer-readable storage medium of claim 46, wherein the method further comprises detecting a location where the instance is being processed.

59. (Previously Presented) The computer-readable storage medium of claim 46, wherein the method further comprises detecting a location where an instance state is being stored.

60. (Previously Presented) A computer-readable storage medium having computer-executable instructions for performing a method for displaying on a display device a graphical debugging environment for a business process service, the method comprising:

obtaining design information about the business process service;

obtaining configuration information about the business process service;

generating a symbolic representation of the operation of the business process service according to the design information and configuration information; and

displaying on the display device a graphical debugging environment showing the symbolic representation.

61. (Previously Presented) The computer-readable storage medium of claim 60, wherein the method further comprises receiving runtime data for the business process service and presenting the runtime data on the display device.

62. (Previously Presented) The computer-readable storage medium of claim 61, wherein the runtime data comprises message flow information.

63. (Previously Presented) The computer-readable storage medium of claim 60, wherein the graphical debugging environment enables a user to place a breakpoint in the symbolic representation of the operation of the business process service.

64. (Previously Presented) The computer-readable storage medium of claim 60, the symbolic representation comprising symbols, wherein the graphical debugging environment also displays information about the symbols.

65. (Previously Presented) The computer-readable storage medium of claim 60, wherein the method further comprises receiving input from an input device to place a break point proximate a symbol, and presenting a symbol representing the break point on the symbolic representation.

DOCKET NO.: MSFT-2763/305222.1 **PATENT**
Application No.: 10/676,743
Office Action Dated: January 16, 2009

REMARKS

In summary, 62 claims numbered 1-20, 23-35, and 37-65 are pending. Claims 1, 12, 26, 40, 46, and 60 are independent. Claims 1, 7, and 12 are hereby amended without adding new matter. The Office Action withdrew previous rejections and their finality, but asserts new grounds of rejection. Claims 1-7, 9-11, 40, 41, 43-45, 60, 61 and 63-65 are rejected under 35 U.S.C. § 102. Claims 8, 12-20, 23-35, 37-39, 42, 46-59, and 62 are rejected under 35 U.S.C. § 103. Applicants respectfully traverse the rejections. Reconsideration in view of the foregoing amendments and following remarks is respectfully requested.

References to the pending application are to the published version of it, i.e., Published Application Publication No. 2005/0071243. Although believed to be unnecessary to overcome the cited references, some amendments were made to clarify claim language or to add additional patentable distinctions over the cited references. Support for indirect communications permitting debugging of a process without prior knowledge where it is running may be found, for example, in paragraphs 70-73 and FIG. 4. Support for using historical or tracked events in a symbolic representation of a process to permit post mortem analysis including recreation of the lifespan and interaction of an instance may be found, for example, in paragraphs 08, 09, 12, 23, 64, 67-69, 76, 77, 95 and 102.

Rejection of Claims 1-7, 9-11, 40, 41, 43-45, 60, 61 and 63-65 under 35 U.S.C. § 102(e)

Claims 1-7, 9-11, 40, 41, 43-45, 60, 61 and 63-65 are rejected under 35 U.S.C. § 102(e) as being unpatentable over U.S. Patent Application Publication No. 2004/0168155, by O'Farrell *et al.* (hereinafter referred to as "O'Farrell"). (Office Action, pp. 2-10). Applicants respectfully traverse the rejection.

It is respectfully submitted that the claimed subject matter is allowable over O'Farrell because patentable distinctions were overlooked, O'Farrell does not teach what it is alleged to teach and additional patentable distinctions are made by present amendments.

First, O'Farrell does not teach or suggest multiple communication methods between the debugger and remote computer running a business process, let alone the two methods in claim 1. Indirect communication, illustrated in one embodiment in FIG. 4 communication path through message box 425, permits debugging process 445 to begin debugging process 405 without knowing machine 400 is running it. Subsequently, a direct communication channel is established between debugging process 445 and business process 420 running on machine 400. Direct communication channels are discussed, for example, in paragraphs 0024, 0064, 0072 and 0075 of the Published Application. O'Farrell discloses neither the indirect communication nor direct channel communication in the claims.

In contrast, as illustrated in O'Farrell's FIG. 9 and discussed in paragraph 0083, O'Farrell requires knowledge of the host computer running a process. As shown in O'Farrell's FIG. 6 and paragraphs 36, 42, 44, 68, 83 and 88, O'Farrell's flow debugger 64 and debug managers 60 only communicate by passing messages back and forth through communication layer 62. Thus, O'Farrell fails to teach or suggest the claimed indirect and direct channel communications.

Second, O'Farrell does not teach or suggest collecting, reading or using historical information. It is respectfully submitted that the Office Action misconstrues passages in O'Farrell.

The first misinterpretation is with regard to hooks. Hooks are used to start and end debug sessions by transferring control between run-time code and a debug manager process. O'Farrell, ¶¶ 0066, 0079. Hooks are discussed in paragraphs 0042-0046 and FIG. 7 of O'Farrell. Hooks are written into runtime code. Hooks check if a debug flag has been set or unset by flow debugger 64. If set, the hook calls the separate debug manager process, which then communicates with flow debugger 64. Thus, paragraphs 0045 and 0055 (cited by the Office Action), which merely mention common placement of hooks in run-time code, do not in any way teach or suggest collecting, reading or using historical "stored state information regarding events."

The second misinterpretation is with regard to tokens. Tokens are not "tracking information about execution of the business process service." Instead, as explained in paragraphs 0032-0034

and 0087, of O'Farrell, tokens are a representation of the process itself. The claim language refers to tracking information during execution of the process. Thus, the Office Action's citation to paragraphs 0033 and 0034 are in error.

Third, even if O'Farrell did collect and read historical stored state information, it does not use it to generate and display a symbolic representation of the operation of the business process. The claimed subject matter bases a symbolic representation on historical stored state information, permitting for example post mortem analysis including recreation of the lifespan and interaction of an instance that failed.

In contrast, O'Farrell only creates its graphical depiction of flow from original design information in tokens; not historical operation data. O'Farrell, ¶¶ 32-34, 87. O'Farrell's paragraph 0043 (cited in the Office Action) merely states that flow debugger 64 receives debug data for presentation on GUI. This doesn't say what the data is or what window in FIGS. 9-17 it is presented in (a textual window or otherwise). This falls far short of teaching or suggesting that flow debugger 64 receives historical stored state information (in addition to current debugging data) and uses the historical stored state information to construct a graphical depiction of a flow. O'Farrell clearly fails to teach or suggest the claimed subject matter.

For at least the foregoing reasons, it is respectfully submitted that the rejection of claims 1-7, 9-11, 40, 41, 43-45, 60, 61 and 63-65 is inaccurate. Accordingly, Applicants respectfully request withdrawal of the rejection.

Rejection of Claims 8, 12-20, 23-35, 37-39, 42, 46-59 and 62 under 35 U.S.C. § 103(a)

Claims 8, 12-20, 23-35, 37-39, 42, 46-59 and 62 are rejected under **35 U.S.C. § 103(a)** as being unpatentable over O'Farrell in view of "BizTalk Unleashed," 1st ed., Feb. **2002, authored by** Adams *et al.* (hereinafter referred to as "Adams"). (Office Action, **pp. 11**-26). Applicants respectfully traverse the rejection.

The foregoing remarks, in whole or in part, apply equally well to the rejection of claims 8, 12-20, 23-35, 37-39, 42, 46-59 and 62. The Office Action makes the same citations and arguments relative to O'Farrell made against claims 1-7, 9-11, 40, 41, 43-45, 60, 61 and 63-65.

The Office Action does not cite Adams to make up for O'Farrell's lack of disclosure detailed above. The Office Action cites Adams with regard to Microsoft BizTalk's interceptor. The pending application points out that it leverages the benefits of existing interceptor tools. However, the existence of interceptor tools, including in Adams, does not teach or suggest the claimed use of interceptor tools relative to debugging. Ignore the word interceptor. The debugging function of causing a server to enter or leave a debugging state if a break point in found in monitored runtime data is not taught or suggested by Adams.

For at least the foregoing reasons, it is respectfully submitted that the rejection of claims 8, 12-20, 23-35, 37-39, 42, 46-59 and 62 is inaccurate. Accordingly, Applicants respectfully request withdrawal of the rejection.

Amendments made herein as well as amendments previously made are without abandonment of subject matter. Applicant expressly reserves the right to, in the pending application or any application related thereto, reintroduce any subject matter removed from the scope of claims by any amendment and introduce any subject matter not present in current or previous claims.

PATENT

DOCKET NO.: MSFT-2763/305222.1
Application No.: 10/676,743
Office Action Dated: January 16, 2009

CONCLUSION

In view of the foregoing remarks and amendments, it is respectfully submitted that this application is in condition for allowance. Reconsideration of this application and an early Notice of Allowance are requested. Applicants desire to hold a telephone interview with the Examiner and his supervisor following their review of this reply.

Date:

Joseph F. Oriti
Registration No. 47,835

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439